コードのダイナミクス(<特集>複雑系の展望-複雑系若手の立場から)

AUTHOR(S):

橋本, 敬

CITATION:

橋本, 敬. コードのダイナミクス(<特集>複雑系の展望-複雑系若手の立場から). 物性研究 1997, 68(1): 4-17

ISSUE DATE:

1997-04-20

URL:

http://hdl.handle.net/2433/96019

コードのダイナミクス

橋本 敬
toshiwo@puneuma.riken.go.jp
理化学研究所 国際フロンティア研究システム
情報処理グループ 情報表現研究室
〒 351-01　埼玉県和光市広沢 2-1

1　序論

様々なシステムを「記号論的」に見た場合、記号の使われ方やその解釈の体系にはそのシステム自体のなんらかのきまりがあるように思える。ここでいう「記号論的」というのは、対象とするシステムについて、物質レベルの実体を捨象し、記号レベルのみを扱い、ある記号が使われた時それに対して起こる反応や発せられる記号の連鎖に着目して対象をみるということであるが、その中でも、解釈、用法の揺らぎ、あるいは相互のずれを通して、システムがどのように動くかを理解しようとする立場である。本論では、記号の使用方やその解釈の体系を「コード」と呼び、人工的なモデルを用い、動的な観点からコードの創発、進化を論じる。

外部からトップダウン的にコードを与える場合、たとえば通信のプロトコルやプログラミング言語仕様では、記号が表す内容というセマンティック・レベルと、記号間の関係というシンタクティック・レベルを分離でき、内的な論理的一貫性に基づいたコードを作ることが可能である。しかし、ここで我々が考えたいのは、コードがシステム内部に自己組織的に現れる場合である。このような場合には、実際の記号使用の場というプラグマティックなレベルを考えることが重要となる。記号がそれ自体では情報とならず、その受け手がどのように解釈するか、ということが考えられなければならない。セマンティック・レベルとシンタクティック・レベルが分離できる場合、同義であるが異なる構造を持つ記号表現が考えられる。これをプラグマティックな場でその記号表現の受け手側から見た場合、その２つの表現は異なる解釈をもたらすことも考えられる。つまり、プラブマティック・レベルを通したセマンティクスとシンタックスの分離不可能性が問題となる。

また、記号の使用者と解釈者が異なるコードにのっとっている場合、複数のコードの相互作用により新たなコードが生まれるといったダイナミクスがある。たとえば、自然言語における ピジン／クレオール語や、細胞内共生を通じて遺伝コードが変化する場合など。

また、外からの規定がない場合に、コードが一貫性を持つものとして常に存在するかどうかも問題である。むしろ、論理的一貫性の綻びを修復しようとする動きが、コードのダイナミクスをもたらすのではないだろうか。あるいは、そのコードにのっとった記号使用やコードを作ることが、コード自体を壊すという、コードの自己否定性が本質的なダイナミクスの源と考えられる場合もある。

記号は外界のなにかを指し示すものであり、その表す対象が意味であると考えられることが多い。たとえば、自然言語では、辞書に様々な語の指示対象が列挙されており、それが語の意味を表すとされる。しかし、実際に指示対象を明示することは不可能であり、われわれは、誰かが話した表現の中の語の意味を常に同定して会話しているのではない。むしろ、ある表現がほかの表

現とどのような関係にあるか、どのような文脈で使用されるのか、といった「記号の用方」が意味を規定していると考えるべきである。「ある言葉 W の意味は？」と問われた場合、「Wっていうのは××のことで…」という説明自体がその言葉の用法の一つなのである。

これを遺伝コードの問題として考えてみよう。遺伝コードは、普通、コドン・トリプレットとアミノ酸の対応の体系のことを指す。それは、記号であるコドンがどのアミノ酸を指し示すかを表すコードである。しかし、この遺伝コードが完全に分かっても、実際のゲノムがどのように発現するのかは分からないことの方が多い。アミノ酸が結合してできるタンパク質の働きは、その立体構造が重要であり、反応過程は不安定で、まわりの環境や直前の反応という文脈に依存するものと考えられる。遺伝コードは、個々の記号単体の指し示すものではなく、「生きている状態」というプラグマティックな状況のなかで、それらがどのように使われているのか、というものとして捉えられるべきである。遺伝コードの進化もそういった状況で起こってきたものであり、やはり、記号の解釈側の存在と、つくられるタンパク質の働きをあらわに考えるべきである。

ここでは、純粋に記号のやりとりの中に現れるコードのダイナミクスに感心があるため、記号の指示対象よりも、むしろ、記号の使用方の体系としてのコードに焦点をあて、言語のコードと遺伝暗号という自己組織的なコードの進化を、下位レベルの要素群の動きのなかに、いかにして秩序構造が生まれるかという点を中心にして行なった研究について述べる。

2 言語コードのダイナミクス

言語のコードを事前に書き下す事はできないが、事後的に言葉の使用について記述する事は可能である。しかし、この記述は常に変化しうる。言語コードはコミュニケーション自体による変化をまぬがれ得ないだろう。文法もまた、コミュニケーションを通して変化していく。統語構造は言語発生の初期からじょじょに複雑化してきたと考えられる。

ここでは、会話主体を何らかの内的ルールを持つエージェントと考え、そのルールを生成文法を用いてモデル化する。このようなエージェント達のなす会話のネットワークにおいて、言葉のよう方についてコードのダイナミクス、文法の複雑化、構造化の過程を見ることにする。

また、エージェントの持つルールを生成文法でモデル化することにより、文法の進化過程と計算能力の進化過程を結びつけて考え、文法が Chomsky 階層の中でどのような過程を経て変化するか、上位の階層へと昇る進化が見られるかという点にも注目する。特に、他のエージェントの話すことを理解するというネットワーク構造のなかで、そして計算時間の有限性がある場合の計算能力の進化過程を、コードの発生と絡めて考えてみよう。

2.1 形式文法システムの会話ゲーム

言語コードの進化を研究するために、会話を行う個体のネットワークを考える[1]。各個体 (エージェント) は形式文法を持つシステムとして形式的に次式で定義される。

$$G_i = (\{S, A, B\}, \{0, 1\}, F_i, S) \ . \tag{1}$$

各エージェントは初期記号 Sから 書き換えルール (F_iの要素) を適用していき、書き換えられた語が終端記号 ($\{0,1\}$) のみになった時点で発話する。語の最大の長さは M とし、それよりも長い語は M以降を切りすてて発話する (ここでは $M = 6$ とする)。適用可能なルールが二つ以上

[1]モデルは概略を記すだけにとどめる。詳細は [1] を参照。なお、得点の定義式 (2) は [1] よりも簡単なものになっている。

ある場合は、ランダムに一つを選択する。書き換えた語に非終端記号 ($\{S, A, B\}$) が残っているが適用可能なルールがない場合は何も発話できないとする。

各エージェントはそれぞれの文法に基づき、発っせられた語の理解を試みる。書き換えルールを発話過程とは逆に適用していき、初期記号 Sに達した時点で理解できたとする。ここでは、理解に関する計算時間に制限を加えるため、書き換えのステップが500を越えると、理解できなかったとみなす。

P個のエージェントの存在するネットワークで、語の発話・受理に応じて得点が与えられるというゲームを行う (ここでは $P = 10$ に固定する)。各エージェントが1単位時間内に R回発話するチャンスが与えられる。あるエージェントが1単位時間に得る得点は、発話した語、理解した語、理解された語に関する得点の重みつき平均として、

$$p^{\mathrm{tot}} = \frac{1}{R}\{r_{\mathrm{sp}}(s^{\mathrm{sp}} - f^{\mathrm{sp}}) + r_{\mathrm{rec}}(s^{\mathrm{rec}} \sum_{recog} \frac{1}{step} - f^{\mathrm{rec}}) + r_{\mathrm{br}}\frac{s^{\mathrm{br}} - f^{\mathrm{br}}}{P}\} \tag{2}$$

と定義する。s^{sp}、s^{rec}、s^{br}はそれぞれ、発話した回数、理解した回数、発話した語が理解された回数、f^{sp}、f^{rec}、f^{br} はそれぞれ、発話しない回数、理解しない回数、発話した語が理解されない回数、$step$ は理解するまでにかかった書き換えのステップ数で、$\sum_{recog}$は、理解した時のみの和をとる。

一部のエージェントが得点に応じて新しいエージェントに置き換えられるという、進化ダイナミクスを導入する。新しいエージェントの持つ文法は、付加（新たなルールを F_iの最後に付加する)、置換（新たなルールと古いルールを入れ換える)、削除（ルールをひとつ取り除く）の3種類の変異過程により更新される。

あるエージェントの処理する情報量は発した語の長さと理解した語の長さの積で以下のように定義する。

$$f_l = \frac{1}{RP^2}\sum_{c=1}^{R}\sum_{k=1}^{P}|w_{kl}^{\mathrm{rec}}(c-1)|\sum_{m=1}^{P}|w_{lm}^{\mathrm{rec}}(c)| \tag{3}$$

$$|w_{ij}^{\mathrm{rec}}(x)| = \begin{cases} 1 & x = 0 \text{ のとき} \\ |w_{ij}(x)| & x > 0 \text{ で、エージェント } i \text{ の発した語が} \\ & \quad \text{エージェント } j \text{に理解されたとき} \\ 0 & \text{その他} \end{cases} \tag{4}$$

ネットワーク内を流れる情報量は、f_lの平均 ($\langle f\rangle = \sum_{l=1}^{P} f_l/P$) で測る。またエージェントの計算能力は全体の語の中の理解できる語の割合として定義する。

初期状態としてランダムに作った計算能力の低い文法 (チョムスキー階層では正規文法に属する) を持つエージェントをつくる。ネットワーク内を流れる情報量は時間的に増大する傾向にあるが、あまり変化しない部分と急激な発展が交互に現れるという断続平衡的な様相を呈する（図1)。

平衡するところでは、同じ語を発話・理解しあうエージェントの集団 (共語集団) が形成されている。急激に発展する部分では、エージェントの文法の進化が見られる。以下、共語集団の形成と特徴的な文法の進化、それらにともなう文法の構造化について述べる。

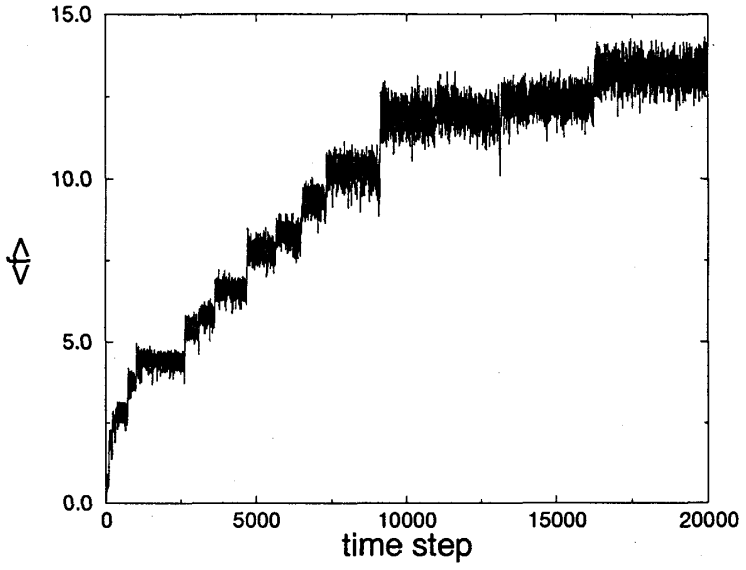

図 1: $\langle f \rangle$ の時間発展。$r_{sp} = 3$、$r_{rec} = 1$、$r_{br} = 1$

2.2 共語集団の形成

時間発展の初期段階では、個々のエージェントの計算能力、発する語の多様性は時間とともに増大していく。それにともない、$\langle f \rangle$ も上昇する。しかし、ある時点で、$\langle f \rangle$ の上昇がとまり、計算能力が高いエージェントがネットワークから取り除かれる場合がある。この時、ネットワーク内ではあるいくつかの語の使用頻度が上がり、これを理解できないエージェントは高い得点が得られなくなっている。これらの語を理解できるエージェントは、お互いにこの語をやり取りすることで、高い得点を得る。この、同じ語を発話・理解しあうエージェントの集団を共語集団 (Ensemble with Common Set of Words, ECW) という。

共語集団が形成されるということは、特定の語を使用することが選ばれており、共通の語の使用という意味のコードが現れていると見ることができる。このコードを共有する集団の中から、次節で述べるような文法の進化により、新しい語を発話できる新たなエージェントが生まれてくる。この新しいエージェントたちが、また新しいコードを持つ集団を形成する。このようにしてコードのダイナミクスが生じ、ネットワーク内の情報量は図 1のような階段状の時間発展を示すのである。

異なる２つの共語集団が存在する時、高い能力のエージェントにより構成される共語集団のメンバーであるエージェントが淘汰される場合がある。より多く話される共語を発話・理解できないエージェントは、たとえ高い計算能力を持っていても淘汰される。共語集団の共存により高い計算能力への進化は抑制されるのである。すなわち、ここでの複数のコードの相互作用では、２つのコードの混合した新たなコードが生まれることは少なく、多数派が少数派を淘汰することが多い。

r_{br}の値が r_{sp}、 r_{rec}の値より非常に大きい場合、すなわち理解しあう事が非常に有利である場合は、一つのコードができるがそれが安定に存在し、ダイナミクスはない。また、語の多様性とネットワーク内の情報は低いレベルに留まる。逆の状況では、語の多様性は上がるが、情報の流れは低い。これはコードが成立せず、相互理解なしに多くの語が発せられている状態である。

2.3 文法の進化

初期状態として最も単純な文法をつくるが、これは分岐のない順次構造をしており、発話できる語数は1語だけである。その後、分岐構造、多重分岐構造の文法を持つエージェントが現れ、エージェントの計算能力は上昇していく。この発展は緩やかなものであるが、まれに急激な発展をみせるエージェントが出現する。この急激な発展は2種類の機構によりもたらされる。

一つはモジュール型進化である。これは、一つの新しいルールをモジュール的に使う事で、多くの新しい文を発話・理解できるようになる発展過程である。自然言語における、接辞による語形成に対応する過程とみなされる。

もう一方はループ構造の出現である。文法内にループを持つことで再帰的な書き換えが可能になり、原理的には無限個の語を発することができる。これは文法の構造的な変化であり、アルゴリズム的進化と呼ぶ。これは、複文を作る事により一つの文で長く豊富な記述ができることに対応する。ループを含む文法のクラスはChomsky階層において文脈自由のクラスと同等であり、初期の正規文法より一つ上のクラスである。すなわち、Chomsky階層をのぼる形の進化が起きている。

共語集団の形成による進化の抑制と、これらの文法の進化が交互に起きることで、断続平衡的な進化が見られる[2]。

2.4 文法の構造化としての二重分節

共語集団の形成と文法の進化を経て、エージェントの持つ文法は構造化されていく[2]。エージェントの持つ文法をみると、

$$\text{非終端記号} \rightarrow \text{終端記号列} \tag{5}$$

$$\begin{aligned} \text{非終端記号} &\rightarrow \text{非終端記号列} \\ \text{非終端記号} &\rightarrow \text{終端記号と非終端記号の組み合わせ} \end{aligned} \tag{6}$$

の2つの形のルールがある。式(5)の形をしたルールは共語集団内でよく話される語を作っている場合が多い。共語集団においてよく話される語はより速く理解できたほうが有利なので、そのような語を受理する過程ができるだけ短くなるような文法になる。つまりよく話される語は、式(5)の形のルールの特別な場合である

$$S \rightarrow \text{よく話される語} \tag{7}$$

というかたちのルールによって理解する。あまり話されない語は、式(6)の形のルールを用いて式(5)の形のルールで作られた語を組み合わすことにより理解しようとする。

これは自然言語の「二重分節」に対応するとみることができる。二重分節とは、アルファベットや五十音のような有限個の記号を用いる自然言語で、無限に多様な文をつくることを可能にしている仕組みのことである。一つの事物に一つの記号を割り当てていれば記号はかなり大きな数が必要であるが、そうするのではなく記号を組み合わせて「語」をつくり、その語をさらに組み合わせて「文」をつくるという仕組みにより多様でかつ構造のある文が作られる。

[2]得点の定義式(2)では理解にかかる書き換えステップ数を考慮している。これを得点に反映させないとき、文法の進化、語の多様性の発展は起きるが、断続平衡的な進化は観察されなかった。

このモデルの結果では、前者の形のルールは「単語」をコードし、後者のルールは「単語」を組み合わせて「文」を作っていると見ることができる。共語集団という上位の秩序構造が造られることにより、それを反映した形でエージェントの持つ文法も構造化してくるのである。

しかし共語集団内では発せられる語が特殊化し、多様な語が話されにくくなりエージェントの能力は低くなってしまうかも知れない。つまり全ての語を 式(7) の形で持てばあらゆる語をもっとも速く理解できるだろう。 これはChomsky 階層では正則文法になる。実際はそうならずに少ないルール数でより多くの語を発話・受理するために、モジュール構造、ループ構造を造る。より多くの語を発話・理解する方向への進化と、共語集団により保たれる文法 (net-grammar [1]) という拮抗は、他のエージェントが話す語をどれだけ理解できるか、あるエージェントの話した語を他のエージェントが理解できるかということが大事であるような多対多のネットワークであるがゆえに生じるものである。もし、ひとつのエージェントのみによって値が決まるような評価関数をもちいて得点を付けたならばこのような拮抗は生じえないであろう。これは、言い換えると外部に固定された環境があるわけではなく自分自身も含めてそこにいるエージェントが環境を決定するということである。他にどのようなエージェントが存在するかにより環境が変化し、さらに自分が発した語により環境が変化すると考えられる。

2.5 文脈依存性と内部モデルの構造化

これまで述べてきたモデルでは、共語の使用というコードの出現、モジュール進化、ループ構造の出現という過程を含む文法の進化、それらの結果としての二重分節への文法の構造化が見られた。しかし、このモデルの範囲では冒頭で述べたような「語の用法としての意味」を表わすには不十分である。なぜなら、各々の語の関係や語と文の間の関係という文脈依存性が明確には扱われないからである。

エージェントの持つ文法を、Chomsky 階層において、 文脈依存文法の一部を含むような階層へと上昇させるために、文法に対する制限をゆるめる事も考えられる。これによりひとつの文の中の語の関係が考慮されるが、文の間の関係はやはり明確には捉え切れない。また、計算量的な問題も深刻になるであろう。

そこで、文法のさらなる複雑化とは少し違った観点から上記の問題に対して取り組んでみよう。ここで、鍵となる概念は「語は文の中で使われる」、「文は語から構成される」という、語と文の相互依存性である。この関係から、語や文の関係を、語、文の作る距離空間内へマッピングする。この相互依存性と語、文の距離空間を通して、 語の間の関係の構造、 文の間の関係の構造を見ることで、より「用法としての語の意味」を明確に表わすことを考える。

この、 語と文の相互依存性というアイデアは Karov and Edelman[3] によるコーパスの中での多義語の明確化のために提出されたものであるが、ここで我々が注目したいのは、コーパスの中という静的なものではなく、エージェント間の会話を通して、動的にエージェントの内的な語や文の関係構造が組織化されてくる過程である。

まず、Karov and Edelman にのっとって語間、文間の距離の定義と計算アルゴリズムを与える。基本的な考え方は、「近い語というのは同じような文に現れる」、また「近い文というのは同じような語で構成される」という相互依存性で、語の距離は文の距離から計算され、逆に文の距離は語の距離から計算される。これを再帰的に繰り返し距離の変化が収束した時点で計算を止める。

具体的には、ある文 s_i と s_j の間の距離、および、ある語 w_i と w_j のあいだの距離を相互再帰的

に以下のように定義する。

$$\mathrm{sim}_{n+1}(s_i, s_j) = \sum_{w \in s_i} \mathrm{weight}(w, s_i) \max_{w_k \in s_j} \mathrm{sim}_n(w, w_k) \tag{8}$$

$$\mathrm{sim}_{n+1}(w_i, w_j) = \sum_{s \ni w_i} \mathrm{weight}(s, w_i) \max_{s_l \ni w_j} \mathrm{sim}_n(s, s_l) \tag{9}$$

ここで添え字 n は、 iteration の回数を表わす。$\mathrm{weight}(w, s_i)$、$\mathrm{weight}(s, w_i)$ は規格化因子である。また、$w \in s$ は文 s に含まれる単語を、$s \ni w$は語 wを含む文を表す。

式 (1) で定義されるエージェントはその文法に基づいて聞いた語を解析し、どういった語の組み合わせであるかを判断する。 たとえば、 $S \to A0B, A \to 10, B \to 11$ という文法を持つエージェントは “10011” という文を聞いたとき、図 2(a) のように解析し、 “10・0・11” という語の並びとして理解したとみなす。この解析結果をもとにして、いままでの語の距離のマトリックスを更新する。

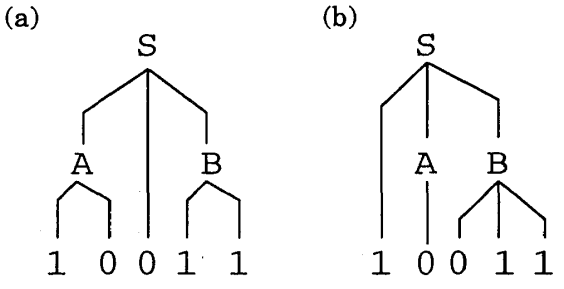

図 2: 異なるエージェントの解析木の例

たくさんの文を聞いているうちに、エージェントの内部の語の距離空間と文の距離空間（これらをあわせて言語空間と呼ぶことにする）が組織化されてくる。そのなかでは、いくつかの語が比較的近い距離で局在し、少しはなれてまた別の語群が局在しているという組織化の状態が生じるであろうが、これは、語の距離に応じたカテゴリー化が起きている状態である。あるいは語の使われ方によって、動詞的な働きと名詞的な働きが分化してくることも考えられる。ある文を通して、異なるカテゴリーの領域が結びつく状況は、メタファーの形成と考えられるであろう。また、発話過程にこの言語空間の構造を反映させることにより、エージェント特有の語の用法が現れてくる。これは、エージェントの内部モデルに相当するものと言える。

また、2 つ以上のエージェントの会話を考えた場合には、それぞれの言語空間の組織化の相互作用が観察される。持っている文法が違えば同じ文に対する解析が異なる。たとえば、$S \to 1AB, A \to 0, B \to 011$ という文法を持つエージェントは “10011” という文を図 2(b) のように解析し “1・0・011” という語の並びとして理解する。このエージェントと上記のエージェントが会話をする場合、同じ文を聞いているにもかかわらず、その会話を通して構成されてくる内部の言語空間は異なったものになる。

複数のエージェント間で共有される言語空間の構造は、また一つの共有コードとなるであろう。逆に、一方にとっては言語空間をそれほど変化させない発話が、別のエージェントの言語空間を大きく変化させることになるような文はコードのずれによる内部モデルのダイナミクスの理解に重要である。

このように、抽象的な言語空間が会話を通して組織化されて行くダイナミクスにより言語コードの進化を論じることが、これまで述べてきたコードのダイナミクスの概念に新たな発展をもたらすであろう。

3 遺伝コードのダイナミクス

この章では、自己触媒ネットワークの進化から自己複製を可能にするコードについて考えてみよう。自己複製には自己言及のパラドックスがある。つまり、自己複製のために自己を観測せねばならず、観測に対して安定でないものの複製は困難となる。von Neumann はマシンとその記述テープ、および記述テープを（解釈せずに）コピーするマシンを組にすることにより、自己複製オートマトンを構成した[4]。しかし、継続的な複製には記述テープ上の情報の安定性が問題になる。また、自己複製マシンがたくさん相互作用する状況では読み方の混合 (syntactic mixture) や 読んだ時の意味のとり方の混合 (semantic mixture) が起こり得る。重要なことはいかにして自己の記述を獲得するかであるが、Eigen らは単体として自己複製するのではなく、お互いにお互いをつくることを助けることで、全体として自己複製するようなハイパーサイクルにより、情報の安定性の面が解決できることを示した[5]。しかし、ハイパーサイクルは自分自身を複製する要素が出現した場合や、ある要素がネットワーク内の他の要素の複製を助けるようになった場合に、常に短いネットワークだけが行き残ってしまう、またある要素の濃度が0になった場合にネットワーク全体が消滅してしまう、そして、非一様な結合が存在できない、といった困難が指摘されている[6, 7]。

ここでは、記述テープとそれを解釈するマシンのアンサンブルを考え、それが自己触媒ネットワークを作ることができるか、その時の「コード」はどのようになるのかを議論する。また、上で述べたようなハイパーサイクルの困難が解決されるかどうかも焦点のひとつである。

3.1 TM ネット – テープとマシンの共進化システム

自己複製を可能にする遺伝コードがどのように進化するかを見るために、ここでは、テープとそれを読むマシンのネットワークを考える[3]。テープは円形のビット列で、マシンの記述となっている。マシンは head パターン、tail パターン、および transition table から構成される。マシンはテープ上に head と tail の両方のパターンを異なる位置に見つけたときのみ、そのテープと反応する。そして、transition table に基づいてテープを解釈し、新しいテープとマシンをつくる。この反応に応じて、それぞれのテープとマシンのポピュレーションは時間発展する。

マシンによってテープが書き換えられる事を「能動的ミューテーション」という。また外部ノイズにより確率的にテープ上の記号が変化することは「受動的ミューテーション」と呼ばれる。ここでは、受動的ミューテーションは、マシンのテープ記号の書き間違いという形で定義する。すなわち、例えば、transition table に基づくならば、本来"0" を書くべきところを、"1" を書いてしまう、というエラーである。

このシステムのアトラクターは、大きく分けて、全てのテープとマシンがなくなる全滅状態 ("Extinction")、一つのテープとマシンの対が存在する単純な自己複製状態 ("Simple Self-replicate")、たくさんのマシン、テープがお互いに作り合うコアネットの状態に分類できる。コアネットはさらに、固定点コアネット ("Fixed Core-net") と振動コアネット ("Oscillatory Core-net") に

[3] モデルは概略のみを示す。詳細は [8] を参照

分かれる。外部ノイズの大きさ (μ_P) に対する各アトラクターへ行くベイシンのボリュームは図3のようになる。

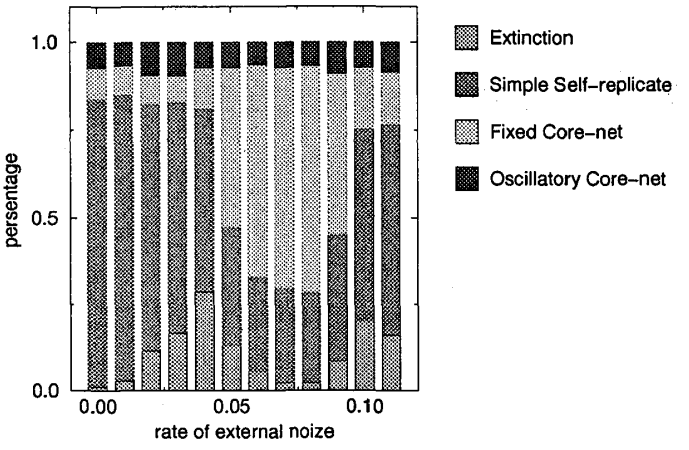

図 3: 外部ノイズに対するベイシンボリュームの変化

3.2 固定点コアネットへの進化

図3からわかるように、外部ノイズが低い場合は、能動的ミューテーションなしで自己複製する、一対のテープとマシンによる最小のネットワークのベイシンが大きい。受動的ミューテーションによりつくられたマシン、テープが最小のネットワークにパラサイト的につながっていきネットワークが大きくなるが、あまり大きなネットワークには成長せず、また元の最小のテープとマシンの対に戻る。外部ノイズが低い時 ($0 \leq \mu_P < 0.05$) は、これを時間的に繰り返す場合が多い。しかし、外部ノイズの大きさの上昇により、単純な自己複製ネットが壊れ、絶滅する場合が増える。

外部ノイズがもう少し大きい領域 ($0.05 \leq \mu_P < 0.10$) では、絶滅、単純な自己複製が少なくなり、固定点コアネットのベイシンが大きくなる。ここでは、最小自己複製ネットワークが受動的ミューテーションにより壊れるのではなく、大きな複製ネットワークへと進化する。受動的ミューテーションにより作られる反応パスが、能動的ミューテーションに置き換えられ、外部ノイズなしで安定に自己維持するネットワーク（コアネット）が出現する。これは個々のテープ・マシンの自己複製から、大きなネットワークが全体として複製する状態への進化であり、このネットワークにおいては、能動的ミューテーション率 (μ_A) が高い値に保たれる。μ_Aの値はマシンがテープを書き換える率である。これを計算を行っている程度と見ると、おおきなネットワークの維持のために計算が行われているという見方が可能である。

固定点コアネットの例を図4に示す。この図を見れば分かるように、大きな自己触媒ネットワークと小さい自己触媒ネットワークが共存している。また非一様な結合も存在できる。

コアネットには幾つかの自己触媒的なネットワークが埋め込まれている。自己触媒ネットワークには、テープが個々に複製されているものと、そうでないものという2つの種類がある。前者は、アイゲンらのハイパーサイクルと同じものであり、$\mu_A = 0$ である。後者はテープとマシン

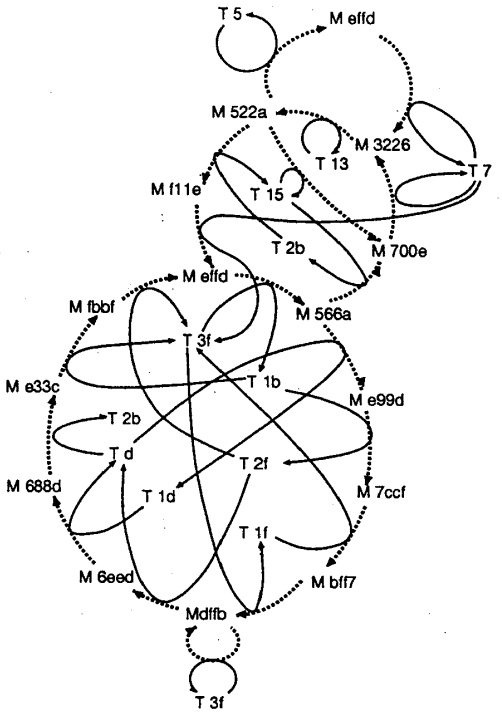

図 4: 固定点コアネットの例

の両方が相互につくりあうループをなしており、$\mu_A > 0$ である。 このタイプでは、非等方的な結合をもつ大きな自己触媒ネットワークが安定に存在できる。

このシステムにおけるコードを考えてみよう。ここでは、テープは遺伝子に、マシンは遺伝子から作られるタンパク質に対応するが、このマシンはまたテープを読む役割も担っている。すなわち、テープがどのマシンに読まれ、どのマシンの設計図として解釈されるかの体系がここでの遺伝コードに対応するものだといえる。図 4に示した固定点コアネットについて、コードを書き下すと表 1のようになる。

このシステムでは同じテープでも読むマシンによって解釈方法がことなるため、多義的なテープが存在する。 この例では $\mathbf{T_7}$、$\mathbf{T_{3f}}$、$\mathbf{T_d}$、$\mathbf{T_{2f}}$、$\mathbf{T_{1b}}$が多義的テープである. これらのテープはそれぞれ二つあるいは三つのマシンによって、二つの異なるマシンとして解釈される。また、 同義テープも存在する。$\mathbf{M_{effd}}$ は $\mathbf{T_7}$、$\mathbf{T_{2f}}$、$\mathbf{T_5}$からつくられるので、このマシンに関しては同義的になっていると言える。

また、非常に興味ある現象として、固定点コアにおいては情報の縮約がみられる。ある固定点コアにおいて、 $\mathbf{T_{3f}}$以外の全てのテープの濃度を 0 にしても、 すぐに元の状態へ復帰するという現象が観察されるのである。この時存在するマシンはこのテープを様々な読み方をすることで、多くの情報を引き出し、ネットワークを元の状態へと戻すのである。 すなわち、 $\mathbf{T_{3f}}$はこのコアネット全体の情報が含まれたキーテープである。

3.3 振動コアネット

振動するコアネットでは、μ_Aやマシン、テープのポピュレーションが振動するだけではなく、ネットワークのトポロジー自体が時間的に変動する。ある時は少数の小さな自己触媒ループでネッ

表 1: 固定点コアネットにおけるコードの例。あるマシン **M** がテープ **T** を読み、そのテープをマシン **M′** として解釈する。**T**、**M** に付けられている添字はテープ、マシンに対して割り振られた番号である。

T	M	M'	T	M	M'
7	effd	3226	3f	dffb	dffb
	700e			bff7	
	f11e	effd		effd	566a
15	522a	700e	d	688d	e33c
	566a			566a	e99d
2f	dffb	6eed	1b	e33c	fbbf
	fbbf	effd		e99d	7ccf
5	522a	effd	13	3226	522a
1d	6eed	688d	2b	522a	f11e

トワークが構成されるが、その後大きなループが多数存在するようになり、また少数の小さなループに戻るという時間的な変動を周期的（あるいは準周期的）に繰り返す。図 3 を見ると、振動コアネットはμ_Pの大きさによらず、ほぼ一定量存在することがわかる。

固定点のコアネットではコードが組織化され安定に存在しているのだが、振動コアの場合はコード自体が時間的に変動する。これは次節で見るように「閉じたコード」をつくれていないと見れる (openness of code)。すなわちあるコードに基づいた解釈をする事自体がネットワークを不安定化させてるというコードの自己否定性がある。

3.4　計算するシステムに対する揺らぎとコードの自己否定性

このシステムは、テープとマシンで構成されるが、それらがなんらかの計算を行ない、システムの維持やダイナミクスをもたらしていると見ることができる。このような、計算するシステムに対する揺らぎとはどのようなものであろうか。たとえば、“3 + 5 = 8” という計算式の 3 が 3.01 に変わることによって、結果が 8.01 になる、というかたちの揺らぎではなく、“3 × 5 = 15” のように、“+” を “×” に、すなわち何を計算するのか、ということを変えてしまうような揺らぎが考えられる。このような計算システムの揺らがせ方をした時に、システムのダイナミクスはどのように変わるかを見てみよう。

TM ネットにおいては、「書き間違い」という受動的ミューテーションはそういった計算システムに対する揺らぎのひとつである。本来、マシンの transition table に従うならば、テープに”1” を書くはずのところを、間違って”0” を書いてしまう。そうすると、そのミューテーションによりつくられたテープからできるマシンは、本来できるはずのマシンとは異なる構成になり、反応できるテープ、そして、何を計算するのかが異なったものになる。「コード」の文脈で言うならば、テープの解釈方法が異なったものが生まれる。このようなかたちの揺らぎがシステムに与える影響がどのようになるかを見たところ、能動的ミューテーションやマシンの数の振動状態から判断されるアトラクターの状態に大きな変化は現れない。すなわち、ネットワークの状態は大きくは

変化しないことがわかる。しかし、周期の変化はよく観察される。

このシステムは、テープとマシンの反応ルールと、それに基づいたポピュレーション・ダイナミクスによって時間発展する。すなわち、反応のルールはポピュレーション・ダイナミクスの遷移行列を決定するメタ・ルールとして働く。コアネットができている状態とは、この遷移行列が決まった状態である。「書き間違い」という揺らぎによっては、このメタ・ルール・レベルでの変化はそれほど起きず、ポピュレーション・ダイナミクスのレベルでの変化しか引き起こされないことがわかった。

このモデルではテープは円環になっている。そして、マシンがheadパターンを探しはじめる位置として、テープ上でのsourceのサイトが指定されている。この位置を変更することも計算システムに対する揺らぎの一つと考えられる。すなわち、テープのsourceの位置の変更は、あるテープがどのマシンと反応できるか、反応後にどのようなものができるかが変更されるので、テープのマシンに対する解釈のされ方（「コード」）を変えることになる。

一例として、固定点コアネットの節で述べた、ある固定点コアにおいて、全体の情報が集約されているテープ $\mathbf{T_{3f}}$ のsourceの位置を変えた時のアトラクターの変化を見てみることにしよう。この場合、$\mathbf{T_{3f}}$ のsourceの位置を変えることによって、図5のように、固定点コアと3種類の振動コアの間を遷移する。テープのsource位置の変更は、エフェクティブに遷移行列を変化させることになるので、あるアトラクターにおいて重要なテープのsource位置の変更により、このようにドラスティックに力学状態が変わることになる。

この例において、固定点コアから振動コアに変わる場合の $\mathbf{T_{3f}}$ に関するコードを詳しくみると、振動時のsource位置での $\mathbf{T_{3f}}$ の解釈では、$\mathbf{T_{3f}}$ 自体をあまり作らないようなコードになっている。つまり、コードが自己否定的になっているのである。その後、他のテープから $\mathbf{T_{3f}}$ が作られるのであるが、主要なテープが自己否定的なコードになる場合は、全体として自己維持できるような閉じたコードができにくいのでコード自体が振動するというダイナミクスとして現れるのである。

4　むすび

言語的コードの進化の研究において、ある語を共通に使う集団が現れた。そこでは、語の使い方が選ばれるというコードが共有されている。また、文法のモジュール型進化、ループ構造の出現により、Chomsky階層をのぼるかたちで高い計算能力へと進化し、コードにダイナミクスが生じる。その結果として、ネットワーク内の情報量は断続平衡的な発展を見せる。

遺伝的コードの進化の研究においては、多くのテープとマシンが相互につくりあう事により、大きなネットワークが全体として自己複製する「コアネット」への進化が見られた。そこでは、マシンによるテープの解釈という、遺伝コードに対応するものが組織化されている。また。ネットワークのトポロジーが時間的に変動する振動コアネットができるが、これは自己否定的なコードによって生み出される振動とみることができるだろう。

現在自然界に見られるコード、たとえば、遺伝コードや自然言語、あるいは、脳における情報処理過程やホルモンによる細胞間情報伝達のしくみがどのようになっているかを考える場合、本来ならば、様々な物理的制約や身体性を考慮すべきであろう。しかし、ここでは具体的な制約をかさず、抽象的なレベルでの記号のやりとりの中に生まれる自発的な構造とそのダイナミクスに焦点をあてている。この、コードのないところからあるところへ、という進化的な見方により、そのシステムなりの制約を反映した形のコードの出現を見ることができる。今後、「そのシステムな

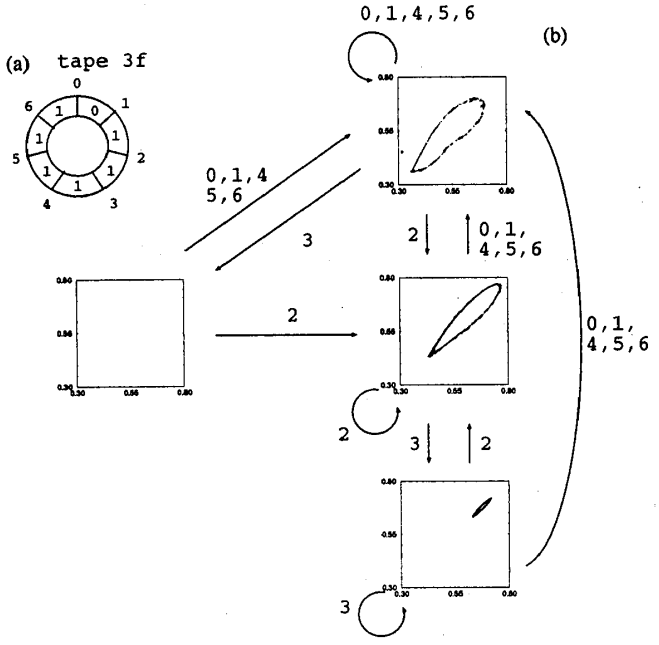

図 5: テープの source の変化によるアトラクター間の転移：(a) テープ 3f 上の記号（枠内）と source 番号（枠外）、(b) source を変えた時のアトラクター間の移り変わり、矢印の横の数字は source 番号を表す。グラフは平均能動的ミューテーション率のリターンマップである。

りの制約」のなかにモデル化の対象の制約をより自然に埋め込むことが求められる。

[謝辞]

本研究の一部は科学技術庁基礎科学特別研究員制度により補助を受けました。ここに記した研究はすべて、東京大学教養学部の池上高志氏との共同研究です。本稿を書くことを心良く了承して下さった池上氏に感謝します。§.2.5 でのアイデアは、東大教養学部における、池上高志氏、泰地真弘人氏、佐藤譲氏との議論に大いに触発されました。ここに感謝の意を表します。また、理化学研究所 FR 情報処理グループ甘利俊一リーダーには、常日頃のアドバイスに感謝します。

最後になりましたが、私の原稿が遅れましたことにより、共同編集者である山本知幸氏、山口明宏氏および、他の執筆者の方々に大変御迷惑をおかけいたしましたことをお詫びいたします。

参考文献

[1] Hashimoto, T. and Ikegami, T., (1996), "Emergence of net-grammar in communicating agents", *BioSystems*, **38**,1–14

[2] Hashimoto, T. and Ikegami, T. (1995), "Communication Network of Symbolic Grammar Systems" in *Proceedings of the International Conference on Dynamical Systems and Chaos*, vol. 2, Y. Aizawa et al. (eds.), World Scientific, Singapore, pp. 595–598

[3] Karov, Y. and Edelman, S., (1996) "Similarity-based Word Sense Disambiguation" Te-

chinical Report of Weizmann Institute, CS-TR 96-06

[4] von Neumann, J., (1968) *"Theory of Self-reproduction"*, the University of Illinois Press, Urbana

[5] Eigen, M. and Schuster P., (1979), *"Hypercycle"*, Springer-Verlag, Berlin

[6] Niesert, U., Harnasch, D. and Bresch, C., (1981), "Origin of life between Scylla and Charybdis", J. Mol. Evol., **17**, 348

[7] Kauffman, S. A., (1993), *"The Origins of Order: Self-Organization and Selection in Evolution"*, Oxford University Press, Oxford

[8] Ikegami, T. and Hashimoto, T., (1996), "Active Mutation in Self-Reproducing Networks of Machines and Tapes", *Artificial Life*, **2**, 305–318